## *AMENDMENTS TO THE CLAIMS*

Please amend the claims as indicated hereafter. [Use ~~strikethrough~~ for deleted matter (or double square brackets "[[]]" if the strikethrough is not easily perceivable, *i.e.*, "4" or a punctuation mark) and underlined for added matter.]

21. (Currently amended) A method of providing recovery from an error condition during initialization of a generated program code, comprising the steps of:

calling a generated function code;

testing an assertion in the generated function code, such that when the assertion is ~~true~~ false, the generated function code performs the steps of:

generating a hidden failure code;

returning to the generated program code; and

returning the hidden failure code to the generated program code;

and such that when the assertion is ~~false~~ true, the generated function code performs the steps of:

continuing processing of the generated function code; and

returning to the generated program code when the processing of the generated function code is completed; and

inserting an error recovery code into the generated program code when error recovery is enabled and when the hidden failure code is returned, the step of inserting performed by the generated program code.

22. (Previously presented) The method of claim 21, further comprising the step of resetting the hidden failure code, the step of resetting performed in conjunction with the step of inserting the error recovery code into the generated program code.

23. (Previously presented) The method of claim 21, wherein the step of generating the hidden failure code further comprises generating a hidden failure flag by the generated function code.

24. (Previously presented) The method of claim 21, wherein the step of inserting the error recovery code further comprises determining if error recovery is enabled, the step of determining performed by the generated program code after the step of returning the hidden failure code.

25. (Previously presented) The method of claim 24, wherein the step of determining if error recovery is enabled further comprises the step of aborting when error recovery is disabled.

26. (Previously presented) The method of claim 24, wherein the step of determining if error recovery is enabled further comprises the step of exiting the generated program code when error recovery is disabled.

27. (Previously presented) The method of claim 21, further comprising the step of aborting when error recovery is disabled, the step of aborting performed by the generated program code.

28. (Currently amended) The method of claim 21, wherein the step of testing the assertion in the generated function code further comprises the step of immediately returning to the generated program code when the assertion is ~~true~~ false.

29. (Previously presented) The method of claim 21, further comprising the step of compiling a computer program, wherein the steps of calling, testing, generating and inserting are performed during compiling.

30. (Previously presented) The method of claim 21, further comprising the steps of:

detecting if a call to a subroutine exists in the generated program code; and

creating an error recovery flag test code to test if error recovery is enabled and to test if the subroutine exists.

31. (Previously presented) The method of claim 21, further comprising the step of generating code to conditionally skip a program abort code and an error recovery flag code when the hidden failure code exists and error recovery is not enabled.

32. (Previously presented) The method of claim 21, wherein the step of inserting an error recovery code into the generated program code further comprises the step of inserting code between a #pragma recover if statement and a #pragma recover end statement.

33. (Previously presented) The method of claim 21, wherein the step of inserting the error recovery code into the generated program code further comprises the step of inserting code between a #pragma recover else statement and a #pragma recover end statement.

34. (Previously presented) The method of claim 21, further comprising the step of continuing processing of the generated program code, the step of continuing performed after the step of inserting the error recovery code into the generated program code.

35. (Currently amended) A system for providing recovery from an error condition in a computer program, said error recovery system comprising:

means for calling a generated function code by [[a]] calling a generated program code;

means for testing an assertion in the generated function code, such that when the assertion is ~~true~~ false, the generated function code generates a hidden failure code, returns to the generated program code, and returns the hidden failure code to the generated program code; and such that when the assertion is ~~false~~ true, the generated function code continues processing of the generated function code and returns to the generated program code when the processing of the generated function code is completed;

means for resetting the hidden failure code in conjunction with inserting an [the] error recovery code into the generated program code; and

means for inserting the error recovery code into the generated program code when error recovery is enabled and when the hidden failure code is returned.

36. (Previously presented) The system of claim 35, wherein the means for generating the hidden failure code further comprises means for generating a hidden failure flag.

37. (Previously presented) The system of claim 35, wherein the means for inserting the error recovery code further comprises means for determining if error recovery is enabled, means for inserting operable after the means for testing has returned to the generated program code.

38. (Currently amended) The system of claim [[38]] 37, wherein the means for determining if error recovery is enabled further comprises means for aborting when error recovery is disabled.

39. (Currently amended) The system of claim [[38]] 37, wherein the means for determining if error recovery is enabled further comprises means for exiting the generated program code when error recovery is disabled.

40. (Currently amended) The system of claim 35, further comprising means for aborting when error recovery is disabled, the ~~step of~~ means for aborting performed by the generated program code.

41. (Currently amended) The system of claim 35, wherein the ~~step of~~ means for testing the assertion in the generated function code further comprises ~~the step of~~ means for immediately returning to the generated program code when the assertion is ~~true~~ false.

42. (Currently amended) The system of claim 35, further comprising:

means for detecting if a call to a subroutine exists in the generated program code; and

means for creating an error recovery flag test code to test if error recovery is enabled and to test if the subroutine exists.

43. (Currently amended) A computer-readable medium for providing recovery from an error condition in a generated program code, comprising:

logic that calls a generated function code;

logic that tests an assertion in the generated function code, such that when the assertion is ~~true~~ false, the generated function code generates a hidden failure flag, returns to the generated program code, and returns the hidden failure flag to the generated program code; and such that when the assertion is ~~false~~ true, the generated function code continues processing of the generated function code and returns to the generated program code when the processing of the generated function code is completed;

logic that resets the hidden failure flag in conjunction with inserting an error recovery code into the generated program code; and

logic that inserts the error recovery code into the generated program code when error recovery is enabled and when the hidden failure flag is returned.